1 Quantum Streaming Algorithms

1.1 Classical Streaming for Triangle Counting and Distinguishing

**Triangle Counting**

**Input:** Graph $G = (V, E)$

**Question:** Approximate the number of triangles in $G$.

A related problem that is usually considered in the literature is that of **Triangle Distinguishing**, which is defined as follows.

**Triangle Distinguishing**

**Input:** Graph $G = (V, E)$ and an integer $T$. It is promised that $G$ has either 0 triangles or $T$ triangles.

**Question:** Does $G$ have 0 triangles or at least $T$ triangles?

Notice that one can use one instance of **Triangle Counting** to solve an instance of **Triangle Distinguishing**. Therefore, a lower bound on **Triangle Distinguishing** implies a lower bound on **Triangle Counting**.

Let $n, m$ and $T$ denote the number of vertices, edges and triangles in the input graph $G$. We define $\Delta_V$ (respectively $\Delta_E$) to be the maximum number of triangles in $G$ that share a vertex (respectively an edge).

**NEED DEF OF BOOLEAN HIDDEN MATCHING**

The following are known.

**Theorem 1.**

1. (Jayaram & Kallaugher [5]) There is a single-pass streaming algorithm for TC that uses space $\tilde{O}\left(\frac{m\Delta_E}{T} + \frac{m\sqrt{T\Delta_V}}{T}\right)$.
2. (Clayton & Cao & Gilani) Any single-pass streaming algorithm for TD (and hence for TC) uses space $\Omega\left(\frac{m\Delta_E}{T}\right)$. This proof uses a reduction of INDEX to TD.
3. (Clayton & Cao & Gilani) Any single-pass streaming algorithm for TD (and hence for TC) uses space $\Omega\left(\frac{m\Delta_V}{T}\right)$. This proof uses a reduction of BHM to TD.
4. (Clayton & Cao & Gilani) Any single-pass streaming algorithm for TD (and hence for TC) requires space $\tilde{\Omega}\left(\frac{m\Delta_E}{T} + \frac{m\sqrt{2\Delta_V}}{T}\right)$. This follows from Parts 2 and 3. Note that we now have matching bounds for one-pass streaming algorithms for TC.

1.2 Quantum Streaming for Triangle Counting and Distinguishing

We will not define Quantum Communication or Quantum streaming NEED REFS. We will try to replicate the lower bounds on TD and TC from Theorem 1.

Theorem 1 used that INDEX has communication complexity $\Omega(n)$. Fortunately, Ambainis et al. [1] showed that INDEX also has quantum communication complexity $\Omega(n)$. Hence we have the following analog to Theorem 1.2 by the same proof:

**Theorem 2.** (Clayton & Cao & Gilani) Any single-pass quantum streaming algorithm for TD (and hence for TC) requires space $\Omega\left(\frac{m\Delta_E}{T}\right)$. This proof uses a reduction of INDEX to TD.

Can we do the same for Theorem 1.3. No. Gavinsky et al. [2] showed the following (see also Kerenidis07 [8]).
Theorem 3. The quantum communication complexity of Boolean Hidden Matching is $O(\log n)$.

Since we do not have an analog of Theorem ?? we do not have a non-trivial lower bound for TC or TD in the region where $\Delta_E = O(1)$ and $T = \Omega(n)$. Indeed, there is an quantum streaming algorithm that works well in that region. Kallaugher [6] showed the following.

Theorem 4. Restrict TC to the graphs where $\Delta_E = O(1)$ and $T = \Omega(n)$. There is a single-pass quantum streaming algorithm for TC that uses space $O(n^{2/5})$.

Hence we have a problem where Quantum is provably better than classical.

Open 5. Does there exist a $c$ such that any single-pass quantum streaming algorithm for TC uses space $\Omega(n^c)$.

2 Other results in quantum streaming complexity

Here we list some communication problems that are hard even on quantum computers, which may be helpful in proving the space complexity for quantum streaming problems.

1. Any protocol for the following problem needs $\Omega(n)$ qubits of communication [9].

\textbf{InnerProduct}
\begin{align*}
\text{Input:} & \quad \text{Alice gets a string } x \in \{0, 1\}^n; \ \text{Bob gets another string } y \in \{0, 1\}^n. \\
\text{Question:} & \quad \text{Compute } \sum_{i=1}^n x_i y_i \mod 2.
\end{align*}

2. Any protocol for the following problem needs $\Omega(\sqrt{n})$ qubits of communication [12].

\textbf{Disjointness}
\begin{align*}
\text{Input:} & \quad \text{Alice gets a string } x \in \{0, 1\}^n; \ \text{Bob gets another string } y \in \{0, 1\}^n. \\
\text{Question:} & \quad \text{Compute } \bigvee_{i=1}^n (x_i \land y_i).
\end{align*}

3. Any protocol for the following problem needs $\Omega(n^2 \log p)$ qubits of communication [14].

\textbf{Singularity}(p)
\begin{align*}
\text{Input:} & \quad \text{Alice gets an } n \times n \text{ matrix } X \text{ over } \mathbb{F}_p; \ \text{Bob gets another } n \times n \text{ matrix } Y \text{ over } \mathbb{F}_p. \\
\text{Question:} & \quad \text{Determine whether } X + Y \text{ is singular over } \mathbb{F}_p.
\end{align*}

In the following, we briefly review some hardness results for quantum streaming problems. We begin with the known results about the DYCK(2) problem, defined as follows.

\textbf{DYCK}(2)
\begin{align*}
\text{Input:} & \quad \text{A string } s \text{ over the alphabet } \{a, \bar{a}, b, \bar{b}\}. \ \text{The alphabet can be interpreted as encoding two types of parenthesis where } \bar{a} \text{ is the closing parenthesis for } a \text{ and } \bar{b} \text{ is the closing parenthesis for } b. \\
\text{Question:} & \quad \text{Is } s \text{ well-parenthesized?}
\end{align*}

Jain and Nayak [4] showed that recognizing DYCK(2) is hard if the stream of parentheses is unidirectional:

\textbf{Theorem 6.} For any $p \geq 1$, any unidirectional bounded-error randomized $p$-pass streaming algorithm for DYCK(2) needs space $\Omega\left(\frac{\sqrt{n}}{p}\right)$, where $n$ is the length of the string.

**Theorem 7.** For any \( p \geq 1 \), any unidirectional bounded-error quantum \( p \)-pass streaming algorithm for \( \text{DYCK}(2) \) needs space \( \Omega\left(\frac{\sqrt{n}}{p^p}\right) \), where \( n \) is the length of the string.

Theorems 6 and 7 are based on the communication complexity problem known as **Augmented Index**, introduced in [1] and defined as follows.

**Augmented Index**

**Input:** Alice gets a string \( x \in \{0,1\}^n \); Bob gets an index \( i \in [n] \) and the sub-string of \( x \) comprising of the first \( i - 1 \) characters.

**Question:** Compute \( x_i \), using one-way communication from Alice to Bob.

For constant number of passes \( p \), Theorem 7 is a generalization of Theorem 6. However, it is not known whether one can achieve a quantum protocol that could bypass the randomized lower bound result of Theorem 6 for some \( p = \omega(1) \).

Now, consider the **Frequency Moment**(\( k \)) problem.

**Frequency Moment** (\( k \))

**Input:** A data stream of numbers \( y_1, \ldots, y_n \) from \([m]\). Let \( n_j = |\{\ell : y_\ell = 1\}| \).

**Question:** Compute \( F_k = \begin{cases} \max_j n_j & k = \infty \\ \sum_j n_j^k & \text{otherwise} \end{cases} \).

Montanaro [10] showed quantum streaming algorithms for the **Frequency Moment** (\( k \)) problem with \( k \in \{0,1,2,\infty\} \) that uses \( \omega(1) \) passes to the input stream and beats the classical lower bound given in [15]. Hamoudi and Magniez [3] generalized this result to all \( k \). We formally state the aforementioned classical lower bound [15] and the quantum upper bound [3] as follows.

**Theorem 8.** For any \( k \in \mathbb{N} \cup \{\infty\} \), any \( p \)-pass randomized streaming algorithm that approximates \( F_k \) uses \( \Omega(n^{1-2/k}/p) \) bits of memory.

**Theorem 9.** For any \( k \in \mathbb{N} \cup \{\infty\} \), there is a \( p \)-pass quantum streaming algorithm that approximates \( F_k \) using \( \tilde{O}(n^{1-2/k}/p^2) \) qubits of memory.

Notice that for constant \( p \), the quantum algorithm of Theorem 9 does not have any advantage for the **Frequency Moment** (\( k \)) by Theorem 8. However, for any \( p = \omega(1) \) and for any \( k \in \mathbb{N} \cup \{\infty\} \), this algorithm requires lower memory than the best-known classical algorithm for the **Frequency Moment** (\( k \)) problem with \( k \in \mathbb{N} \setminus \{0\} \cup \{\infty\} \).

### 3 k-Clique Counting and Boolean Hidden Hypermatching

In this section, we define two problems for \( k \)-clique finding which are analogous to **Triangle Counting** and **Triangle Distinguishing**. We also introduce an important problem (**Boolean Hidden Hypermatching**) that we will use to show a lower bound on these problems and we discuss the motivation for this problem.

**k-Clique Counting**

**Input:** Graph \( G = (V,E) \)

**Question:** Approximate the number of cliques of size \( k \) in \( G \).
**k-Clique Distinguishing**

**Input:** Graph $G = (V, E)$ and an integer $C$. It is promised that $G$ has either 0 k-cliques or $C$ k-cliques.

**Question:** Does $G$ have 0 k-cliques or at least $C$ k-cliques?

Like for triangles, one can use an instance of k-Clique Counting to solve an instance of k-Clique Distinguishing. Therefore, a lower bound on k-Clique Distinguishing implies a lower bound on k-Clique Counting. It is usually easier and sufficient to consider lower bounds for k-Clique Distinguishing.

In Theorem ??, we showed a $\Omega \left( \frac{m \Delta E}{T} \right)$ space lower bound Triangle Distinguishing. It is not hard to show that a similar reduction gives the same lower bound for k-Clique Distinguishing; however this gives a trivial lower bound on most graphs, since $\Delta E$ is usually small. We want to show a stronger lower bound for more general graphs. Additionally, since the quantum streaming complexity of triangle counting in the parameter setting $\Delta E = O(1)$ and $T = \Omega(m)$ is an open problem (see section ??), it might be instructive to look for lower bounds on k-Clique Counting for $k > 3$ in this parameter setting to understand if the difficulty of this problem is unique for triangle counting. To do this we will introduce a new problem:

**Boolean Hidden Hypermatching**

**Input:** Alice gets a string $x \in \{0, 1\}^k n$; Bob gets a perfect hyper-matching $M$ over $[kn]$ and a string $w \in \{0, 1\}^n$. $M$ is interpreted as a $n \times kn$ matrix where each row represents a hyper-edge in the matching and $w$ is promised to satisfy either $Mx = w$ or $Mx = \overline{w}$ (where $\overline{w}$ is $w$ with every bit flipped).

**Question:** Determine which is the case: $Mx = w$ or $Mx = \overline{w}$.

Boolean Hidden Hypermatching (BHHM) is a generalization of BHM. Importantly, this problem has a polynomial lower bound both classically and quantumly. The classical lower bound is $\Omega(n^{1-1/k})$ and the quantum lower bound is $\Omega(n^{1-2/k})$ [13].

### 4 Streaming complexity of $k$-clique Counting

**Theorem 10.** Any classical single-pass streaming algorithm for k-Clique Distinguishing requires $\Omega \left( m^{1-1/k} \right)$ bits of space. If the algorithm is allowed to be quantum, it requires $\Omega \left( m^{1-2/k} \right)$ qubits of space.

**Proof.** Assume there is a $o \left( m^{1-1/k} \right)$ space classical streaming algorithm $\text{ALG}_C$ and a $o \left( m^{1-2/k} \right)$ space quantum streaming algorithm $\text{ALG}_Q$ for k-Clique Distinguishing. Note that these work for any arrival order of edges. We use $\text{ALG}_C$ or $\text{ALG}_Q$ in the following protocol for Boolean Hidden Hypermatching. The protocol will take $o(n^{1-1/k})$ space in the classical setting and $o(n^{1-2/k})$ space in the quantum setting, both of which contradict the communication complexity of Boolean Hidden Hypermatching. Hence $\text{ALG}_C$ and $\text{ALG}_Q$ cannot exist. The reduction is the same for both the classical and quantum settings, so we only present it once.

1. Alice has $x_1 \cdots x_{kn} \in \{0, 1\}^{kn}$, and Bob has $M \in \{0, 1\}^{n \times kn}$ and $w_1 \cdots w_n \in \{0, 1\}^n$.

2. Alice and Bob construct different parts of a graph. The vertices are $V = \bigcup_{r=1}^{2k-1} V_r$, where $V_r = \{a^r\} \cup \{b_1^r, \cdots, b_{kn}^r\} \cup \{c_1^r, \cdots, c_{kn}^r\}$. We can think of $V$ as $2^{k-1}$ copies of the graph $V_r$. 

4
Figure 1: Example of a reduction from **Boolean Hidden Hypermatching** to **k-Clique Distinguishing** where $k = 4$, $x = 0110...$, $w = 0...$, and $M$ contains the hyper-edge $(1, 2, 3, 4)$.

- Alice constructs the graph on vertices $V$ by letting

$$E_A = \bigcup_{r=1}^{2^{k-1}} \left[ \{(a^r, b^r_j): x_j = 0\} \cup \{(a^r, c^r_j): x_j = 1\} \right]$$

(1)

Notice that Alice does the same thing on each copy $V_r$ (see Figure 1a).

- Bob constructs the graph on vertices $V$. Let $e_l = (i_1, \ldots, i_k)$ be the $l$'th hyper-edge in the matching $M$. He does the following for each $l \in [n]$:

  - For every $r \in [2^{k-1}]$, he assigns $V_r$ a unique string $y^{(r)}$ of length $k$ with Hamming weight the same parity as $w_l$. So if, e.g., $k = 4$ and $w_l = 0$, he might assign the strings $y^{(1)} = 0000, y^{(2)} = 0011, y^{(3)} = 0101 \ldots$
  
  - For every $r \in [2^{k-1}]$, he fully connects the vertices which Alice would connect to if the bits of $x$ corresponding to $e_l$ were the same as $y^{(r)}$, i.e., the vertices:

$$V_B = \left\{ b^r_{i_j} : i_j \in e_l \land y^{(r)}_j = 0 \right\} \cup \left\{ c^r_{i_j} : i_j \in e_l \land y^{(r)}_j = 1 \right\}$$

(2)

This amounts to Bob drawing $n \cdot 2^{k-1}$ k-cliques (see Figure 1b).

3. (Comment, not part of the algorithm.) See Figure 1 for an example where $k = 4$, $x = 0110...$, $w = 0...$, and $M = \begin{bmatrix} 11110 & \cdots \\ 0000 & \cdots \\ \vdots & \end{bmatrix}$.

4. (Comment, not part of the algorithm.) Note: for each hyperedge $e$, let $x[e]$ be $x$ restricted to the indices in $e$. If $y^{(i)} = x[e]$, then we have a $(k+1)$-clique because Bob draws a $k$-clique and Alice connects vertex $a$ to the same $k$ vertices. On the other hand, if $y^{(i)} \neq x[e]$, then we have no $(k+1)$-cliques because the $k$-clique Bob draws is not fully connected with any other vertices. When $Mx = w$, there will be an $i$ such that $y^{(i)} = x[e]$ so the resulting graph will contain $n$ $(k+1)$-cliques. When $Mx = \overline{w}$, there will be no $i$ such that $y^{(i)} = x[e]$ so the resulting graph will contain no $(k+1)$-cliques.
5. Alice runs $E_A$ through a streaming algorithm ALG (either ALG$_C$ or ALG$_Q$). Since it uses only $o(m^{1-1/k}) = o(n^{1-1/k})$ bits or $o(m^{1-2/k}) = o(n^{1-2/k})$ qubits, when she is done there are $o(n^{1-1/k})$ bits or $o(n^{1-2/k})$ qubits in memory, which she then sends to Bob.

6. Bob continues ALG on his edge set. If ALG returns Yes, then he knows that $Mx = w$; if ALG returns No, then he knows that $Mx = \pi$. The total one-way communication is $o(n^{1-1/k})$ bits or $o(n^{1-2/k})$ qubits.

\[ \square \]

5 Quadratic improvement on the dependence on $k$

The previous reduction from $k$-BHMM to $(k + 1)$-CLIQUE DISTINGUISHING outputs graphs which contain $|V| = \Theta (2^k n)$ number of vertices. Can we improve the dependence of the graph size on the parameter $k$? The answer is yes. We found that we can quadratically reduce the graph size. This is stated as follows.

**Theorem 11.** There is a reduction from $k$-BHMM to $(k + 1)$-CLIQUE DISTINGUISHING on graphs of size $|V| = \Theta (2^{k/2} n)$.

The main idea is to superimpose different “copies” that Bob drew before into a smaller number of “groups”. We will first show how the “copies” are grouped together, and then describe the reduction algorithm.

In the following, we assume $k$ is even for simplicity. For each $y \in \{0,1\}^k$, we compute the parity of every two consecutive bits of $y$ to obtain $\alpha \in \{0,1\}^{k/2}$:

$$\alpha_i := y_{2i-1} \oplus y_{2i}, \quad i \in [k/2],$$

(3)

and we say that $\alpha$ is the name of the group that $y$ belongs to. Note that there are $2^{k/2}$ different groups and each group contains $2^{k/2}$ elements. The following observation is key to our reduction algorithm:

**Lemma 12.** For any $y \in \{0,1\}^k$ and any $\beta \in \{0,1\}^{k/2}$, if $y$ and $\beta$ have different parity in their Hamming weights, then there is an index $i \in [k/2]$ such that for any $z \in \{0,1\}^k$ belonging to the group $\beta$, the following holds:

$$y_{2i-1} \oplus y_{2i} \neq z_{2i-1} \oplus z_{2i}$$

(4)

**Proof.** By way of contradiction, suppose there is no such index $i$. Fix any index $j \in [k/2]$. Then, by assumption, we know that there exists a $z \in \{0,1\}^k$ in the group $\beta$ with $y_{2j-1} \oplus y_{2j} = z_{2j-1} \oplus z_{2j}$. But this by definition implies that $y_{2j-1} \oplus y_{2j} = \beta_j$. Since the above holds for any $j$, we immediately have

$$\bigoplus_{j=1}^{k/2} y_j = \bigoplus_{j=1}^{k/2} (y_{2j-1} \oplus y_{2j}) = \bigoplus_{j=1}^{k/2} \beta_j,$$

which contradicts the assumption that the Hamming weights of $y$ and $\beta$ have different parity. \[ \square \]

With the above grouping construction, we are now in a position to prove Theorem 11.

**Proof of Theorem 11.** We describe the reduction algorithm below:

1. Alice has $x_1 \cdots x_{kn} \in \{0,1\}^{kn}$, and Bob has $M \in \{0,1\}^{n \times kn}$ and $w_1 \cdots w_n \in \{0,1\}^n$.

2. Alice and Bob construct different parts of a graph. The vertices are $V = \bigcup_{r=1}^{2^{k/2}-1} V_r$, where $V_r = \{a_r^r\} \cup \{b_{1r}^r, \ldots, b_{kr}^r\} \cup \{c_{1r}^r, \ldots, c_{kr}^r\}$. We can think of $V$ as $2^{k/2-1}$ copies of the graph $V_r$. 

6
• Alice constructs the graph on vertices \( V \) by letting

\[
E_A = \bigcup_{r=1}^{2^{k/2-1}} \left[ \{(a^r, b^r_j) : x_j = 0\} \cup \{(a^r, c^r_j) : x_j = 1\} \right]
\]

(5)

Notice that Alice does the same thing on each copy \( V_r \).

• Bob constructs the graph on vertices \( V \). Let \( e_l = (i_1, \ldots, i_k) \) be the \( l \)'th hyper-edge in the matching \( M \). He does the following for each \( l \in [n] \):

- For every \( r \in [2^{k/2-1}] \), he assigns \( V_r \) a unique string \( \alpha^{(r)} \) of length \( k/2 \) with Hamming weight the same parity as the bit \( w_l \). So if, e.g., \( k = 4 \) and \( w_l = 0 \), he might assign the strings \( \alpha^{(1)} = 00, \alpha^{(2)} = 11 \).

- For every \( r \in [2^{k/2-1}] \) and every \( z \in \{0, 1\}^k \) in the group \( \alpha^{(r)} \), he fully connects the vertices which Alice would connect to if the \( k \) bits in \( x \) that correspond to the hyper-edge \( e_l \) equal \( z \), i.e., the vertices:

\[
V_B^{r,z} = \left\{ b^r_{i_j} : i_j \in e_l \land z_j = 0 \right\} \cup \left\{ c^r_i : i_j \in e_l \land z_j = 1 \right\}
\]

(6)

This amounts to Bob drawing \( n \cdot 2^{k-1} \) \( k \)-cliques.

3. (Comment, not part of the algorithm.) See Figure 2 for an example of how we can reduce the number of copies from \( 2^{k-1} \) to \( 2^{k/2-1} \).

4. (Comment, not part of the algorithm.) Notice that in the “yes” case \( (Mx = w, \text{ see Figure 2a}) \) where we already had a \((k+1)\)-clique, we still draw the same clique, so the resulting graph will contain \( \geq n \) \((k+1)\)-cliques.

The “no” case \( (Mx = \overline{w}, \text{ see Figure 2b}) \) is more difficult to analyze. First, we notice that the “hub” \( a^r \) must be involved in any \((k+1)\)-clique, since any \((k+1)\)-clique which does not contain \( a^r \) must contain two vertices \( b^r_i \) and \( c^r_i \) (for some \( i \)) by the pigeonhole principle, but \( b^r_i \) and \( c^r_i \) are not connected by construction.

Pick any hyper-edge \( e_l = (j_1, \ldots, j_k) \), denote \( y = x|_{j_1} \cdots j_k \) to be the corresponding bits from \( x \). Fix any “group” \( \alpha^{(r)} \). From Lemma 12, there is an index \( i \) such that for any \( z \in \{0, 1\}^k \) belonging to the “group” \( \alpha^{(r)} \), \( y_{2i-1} \oplus y_{2i} \neq \overline{z}_{2i-1} \oplus \overline{z}_{2i} \). Without loss of generality, assume \( y_{2i-1} \oplus y_{2i} = 0 \). Then, Alice either drew the edges \((a^r, b^r_{j_{2i-1}}), (a^r, b^r_{j_{2i}})\) or she drew the edges \((a^r, c^r_{j_{2i-1}}), (a^r, c^r_{j_{2i}})\). By construction of \( V_B^{r,z} \) in Eq (6), it is easy to see that \( V_B^{r,z} \) contains neither \( \{b^r_{j_{2i-1}}, b^r_{j_{2i}}\} \) nor \( \{c^r_{j_{2i-1}}, c^r_{j_{2i}}\} \). Since this is true for all \( z \) and \( r \), there are no \((k+1)\)-cliques.

\[\square\]

6 Conclusion and open questions

We have shown that \( k \)-clique counting is hard in both the classical and quantum streaming models, proving lower bounds of \( \Omega(n^{1-1/k}) \) and \( \Omega(n^{1-2/k}) \) respectively. Moreover, we showed that the reduction for this bound can be performed with graphs of size \( |V| = \Theta(2^{k/2}n) \).

An important open question is to decide whether there is a polynomial lower bound or a sub-polynomial upper bound for triangle counting in the quantum streaming model for the case when \( \Delta_E = O(1) \) and \( T = \Omega(m) \). Reducing from Boolean Hidden Hypermatching does not seem to work:
(a) \( w_l = 0 \): We are in the “yes” case. The 5-clique is preserved by the grouping.

(b) \( w_l = 1 \): We are in the “no” case. The grouping ensures that there are still no 5-cliques after combining edges.

Figure 2: Example of reducing the number of copies where \( k = 4 \), \( x = 0110... \), and \( M \) contains the hyper-edge \((1, 2, 3, 4)\). The graphs on the left result from the reduction in Section 3 (see Figure 1) We are able to reduce the number of copies from 8 to 2 while maintaining the same number of 5-cliques.

simply assigning \( k = 2 \) in our reduction above gives a vacuous lower bound of \( \Omega(1) \) for triangle counting.

One natural extension of our result would be to determine for what classes of sub-graphs, the quantum streaming complexity of \( k \)-subgraph counting is polynomial in \( n \). In particular, it would be interesting to categorize the quantum streaming complexity of \( k \)-cycle counting and \( k \)-star counting.
References


